Meta Reinforcement Learning for Constant Continuous Reward Tasks

Florentin Hennecker

Université Libre de Bruxelles, Boulevard du Triomphe - CP 212, 1050 Brussels, Belgium
fhenneck@ulb.ac.be

Abstract

Meta reinforcement learning recently emerged as the art of teaching an agent to learn to solve a task as opposed to teaching an agent to solve a task. The generalisation capabilities of meta-learning agents and their very fast learning feats have been tested on multi-armed bandit problems as well as visual navigation tasks. This paper enables the use of a meta-learning setup on continuous state-space problems such as Acrobot and CartPole with a constant timestep-based reward using simple reward shaping. The performance of this setup is tested on a distribution of parametrised Acrobot environments and tampered state observation vectors.

Introduction

There have been incredible advances in the field of reinforcement learning in the past few years. Deep Q-learning (Mnih et al., 2013, 2015) made learning Atari games from visual inputs possible, even beating the human level benchmark for some games; less than two years later, a similar milestone has been reached for playing Doom, a complex 3D game (Lample and Chaplot, 2016) using the VizDoom environment (Kempka et al., 2016). Most of these achievements share one issue: once the agents described in these papers have learned to solve the task they were handed, they are likely to fail on slightly different versions of the task and will have to be completely retrained for new tasks, however similar they are from the task the agent was trained on.

Just like humans use prior knowledge to quickly learn new tasks that are presented to them, training an agent to learn to solve variations of a problem is unlikely to be possible without giving it some sort of prior knowledge. A second key component needed for a fast-learning agent would be a good learning strategy (*in the sense where ϵ-greedy or softmax selection (Sutton and Barto, 1998), Q-learning (Watkins, 1989) and SARSA (Rummery and Niranjan, 1994) are learning strategies*) that takes into account prior knowledge and takes action to learn about the problem as fast as it can to perform optimally as soon as possible.

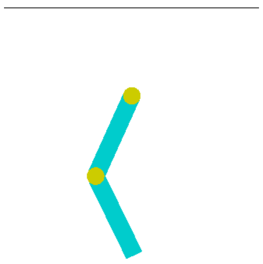


Figure 1: The Acrobot environment

Finding such a strategy is hard, but it can be cast as a reinforcement learning problem: instead of designing a learning strategy by hand and teaching the agent to solve a task, Wang et al. (2016) and Duan et al. (2016) propose to teach an agent to learn to solve a task. The agent presented by Wang et al. is an A2C (a single-threaded version of A3C (Mnih et al., 2016)) agent with a recurrent connection and presents two major differences compared to a standard reinforcement learning agent:

1. the agent is trained on trials of episodes rather than on single episodes, replicating the training process in standard reinforcement learning
2. the agent receives process-level information as well as an observation of the enviromnent: the reward at the previous timestep, the action that led to that reward and an episode termination flag.

Conceptually, in standard reinforcement learning, the network represents a policy and its goal is to collect an optimal reward for one episode. In meta reinforcement learning, the network represents an algorithm which learns a policy. Its weights, once trained, encode a learning strategy which decides how to play episodes to maximise reward across multiple episodes.

Both Wang et al. and Duan et al. show convincing learning behaviour from the meta-learning agent in various types of bandit problems but also a complex visual navigation task (Mirowski et al., 2016). In both situations, the agent deploys

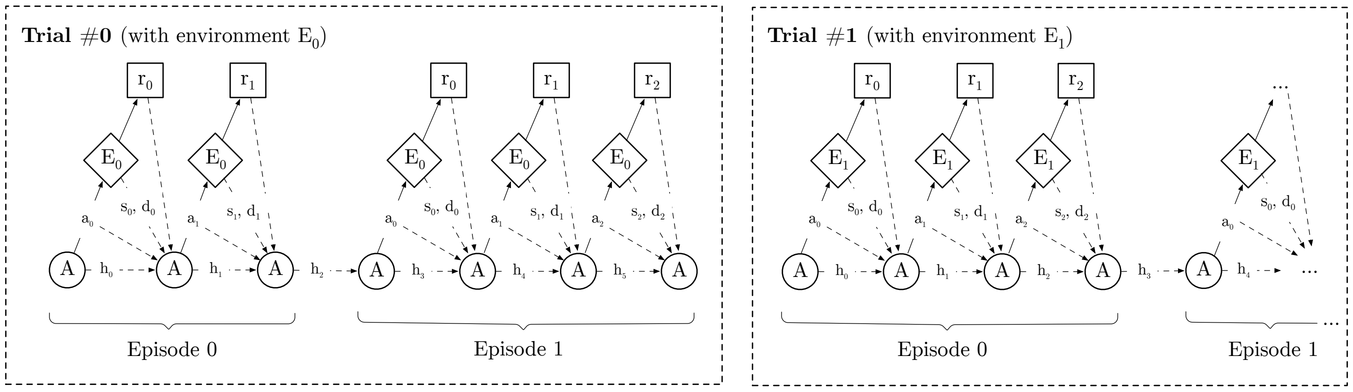

Trial #0 (with environment E0)
Trial #1 (with environment E1)
Episode 0
Episode 1
Episode 0
Episode 1

Figure 2: The process of training a meta-learning agent. A is the agent, E is the environment. At each timestep, it receives the state of the environment s_t, the previous action taken a_{t-1}, the reward of the previous transition r_{t-1} and an episode termination flag d_{t-1} set to 1 is the state is terminal, 0 otherwise.

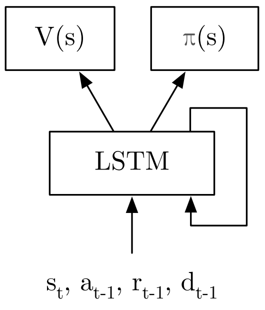


Figure 3: The architecture of the meta-learning agent as presented in Wang et al. (2016) with a LSTM instead of a GRU cell.

a strategy that performs excellent exploration based on its prior knowledge of the structure of the task and then exploits the knowledge gained during exploration to perform optimally for the following episodes. In situations where it is possible, the agent is able to perform one-shot learning.

We will apply meta reinforcement learning to a new class of problems with continuous state spaces and a timestep-based-only reward, generating a distribution of similar tasks by modifying some intrinsic parameters of the Acrobot environment such as the length and mass of the links, but also by tampering with the observation vector obtained by the agent.

Background

The meta-learning A2C agent presented in Wang et al. (2016) is a recurrent neural network with two outputs; a policy output (the actor) and a value output (the critic). Its input is not only the state of the environment s_t, but also the previous action taken a_{t-1}, the reward of the previous transition r_{t-1} and an episode termination flag d_{t-1} set to 1 is the state is terminal, 0 otherwise. Wang et al. (2016) use a GRU (Chung et al., 2014) cell for the recurrent part of the architecture; we use a 48 unit LSTM cell (Hochreiter and Schmidhuber, 1997). Figure 3 shows the architecture of our agent.

The agent is trained to play **trials** of several episodes, and its hidden state is passed along even inbetween different episodes, but not between trials (see Figure 2). Its goal is to maximise its reward across all episodes of a trial.

The loss of the A2C agent is the following :

$$\mathcal{L} = \beta_v \mathcal{L}_v + \beta_p \mathcal{L}_p - \beta_e \mathcal{L}_e$$

where $\mathcal{L}_v$ is the value loss :

$$\mathcal{L}_v = (R_{t-1} - V(s_t))^2$$

$\mathcal{L}_p$ is the policy loss :

$$\mathcal{L}_p = \pi(a_t \mid s_t)(R_t - V(s_t))$$

$\mathcal{L}_e$ is the entropy regularisation :

$$\mathcal{L}_e = \sum_{a \in A} \pi(a \mid s_t) \log(\pi(a \mid s_t))$$

where

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

In our experiments, we use $\gamma = 0.97$, $\beta_v = 0.5$, $\beta_p = 1$, $\beta_e = 0.05$. An update is performed once after every trial using Adam (Kingma and Ba, 2014) with a learning rate of 0.001.

Methods

We will apply the meta-learning setup presented in Wang et al. (2016) and Duan et al. (2016) to a distribution of continuous state problems with a constant reward at each timestep. This distribution will be derived from the original

Acrobot problem (Sutton, 1996; Geramifard et al., 2015). The Acrobot environment (illustrated in Figure 1) consists of two links that are connected with an actuated joint. The upper link is fixed with an unactuated joint. The goal of the agent is to swing the lower link above a given height.

We generate a distribution of similar problems by modifying two of the Acrobot parameters: the link lengths and their masses. The default values for each of these is 1, but for each trial, we will sample uniformly:

- a length $l \in \{0.5, 1, 2\}$ for each link;
- a mass $m \in \{0.1, 1\}$ for each link.

Reward shaping Applying the setup presented by Wang et al. (2016) as is on the Acrobot problem (and other problems with similar reward structures, e.g. the CartPole problem (Barto et al., 1990)) will generate unwanted episode-wise reward dynamics. The reward structure of such problems is simple: either +1 or -1 at every timestep. In human-designed learning strategies, the designer knows that for a problem like Acrobot, a short episode is a good thing, and this knowledge will be implicitly distilled in the learning strategy.

This knowledge is not available to the meta-learning agent. In fact, it does not have any incentive to finish episodes that are far from the trial horizon as quickly as possible. This is due to the fact that the discount factor makes the agent only take into account a limited number of timesteps away from a time t. If the agent is playing the last episode, it can potentially see the end of the constant -1 reward stream and take action to reach it; in the first episodes, whatever the agent does, the negative reward stream will continue no matter how quickly it solved the problem.

This is why we need to introduce a process-level reward, just like we feed process-level information such as the episode termination flag, the reward and previous action taken to the agent. For the Acrobot problem, we can give the agent a small positive reward at the end of an episode to encourage the agent to reach it as quickly as possible. All our experiments feed a reward of +10 at the last timestep of each episode in a trial.

We will compare the performance of an agent trained on trials of 1 episode and on 2 episodes to estimate the impact of letting the agent play one episode after it has interacted with the environment and hopefully estimated its parameters. This comparison will be conducted on different settings, modifying the parameters of the game and introducing unseen behaviours of the environment to evaluate the generalisation performance of both agents.

Experiment 1

The first experiment will compare the performance of our agent in settings that it has seen during training on trials of 1 and 2 episodes. We will only look at the distribution of obtained rewards for final episodes (the second episode in the dual-episode setting and the first and only episode in the single-episode setting) as the nonterminal episodes are considered to be the "learning" episodes.

We fix the agent's weights after training for this experiment. Just to make sure we are clear, we expect the agent to show a learning behaviour after training, meaning that its reward per episode should increase as it plays more episodes. The learning policy should be encoded in the agent's weights.

The agent plays 192 trials for each setting (single-episode and dual-episode). The parameters are sampled uniformly from the same distributions as the training.

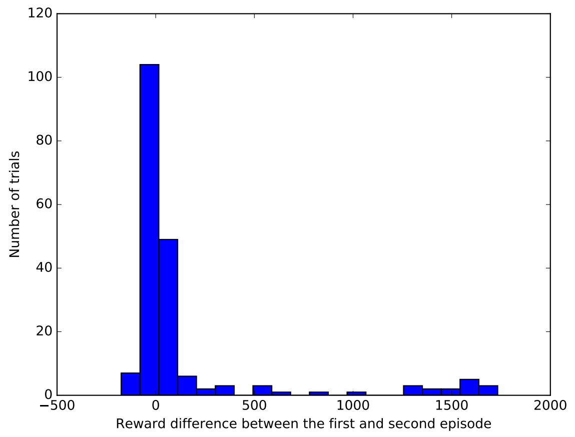


Figure 4: Distribution of the reward difference between the first and the second episode of a dual-episode trial for experiment 1

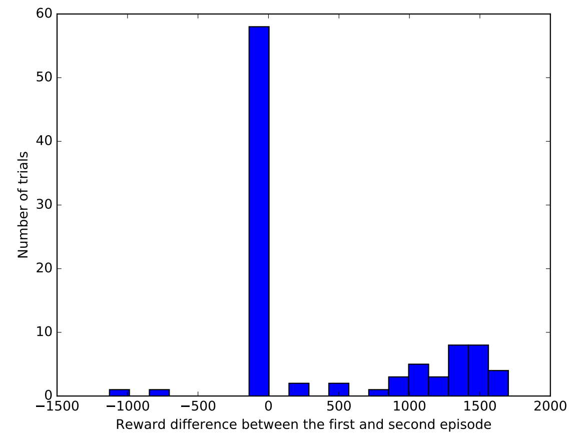
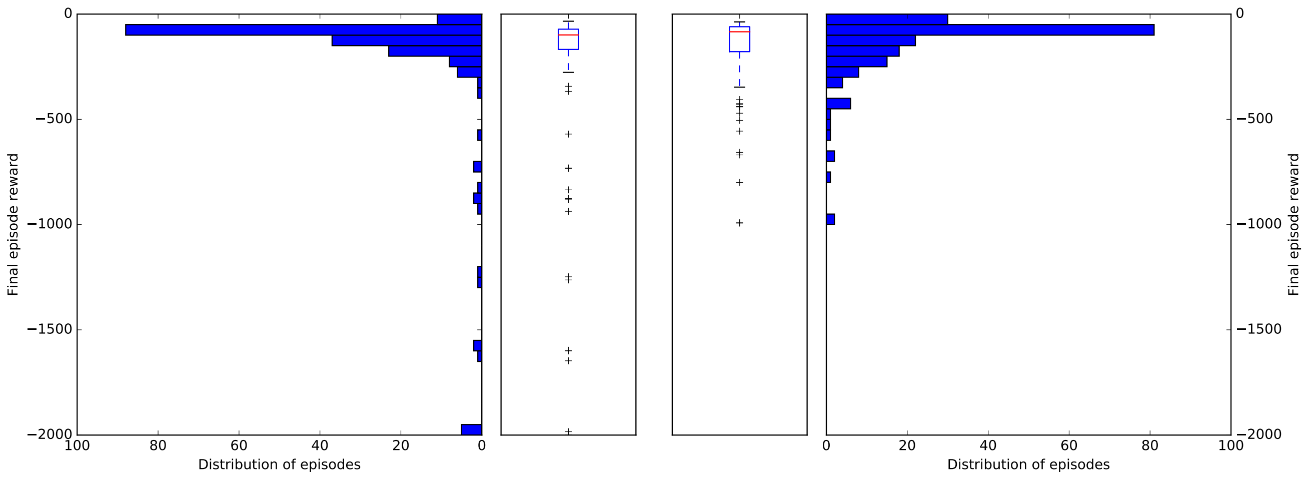


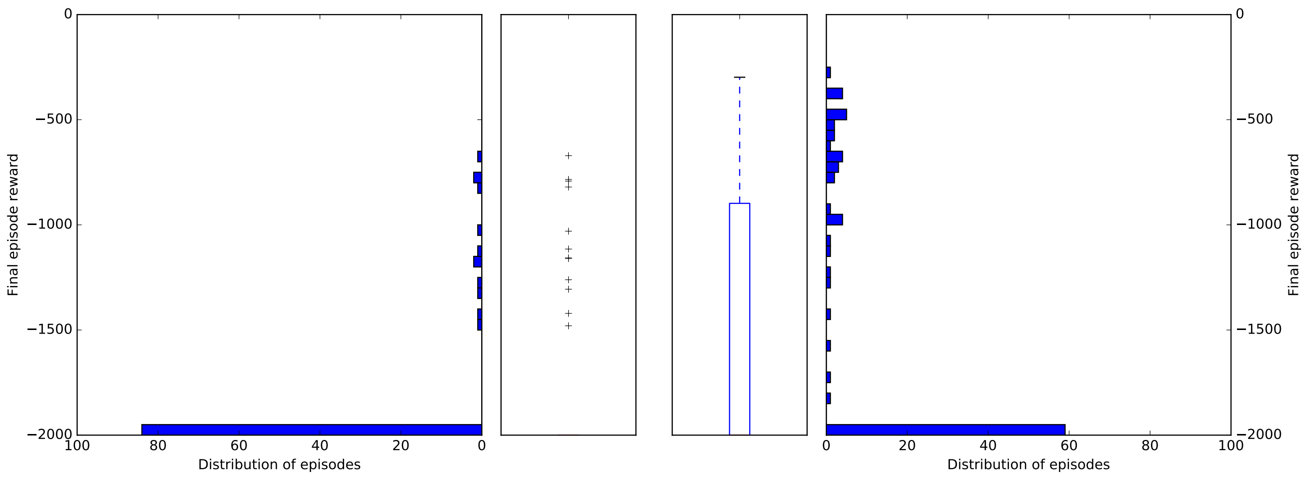
Figure 5: Distribution of the reward difference between the first and the second episode of a dual-episode trial for experiment 2



(a) Trials of 1 episode

(b) Trials of 2 episode

Figure 6: Distribution of the total reward obtained during final episodes for trials of different lengths for experiment 1



(a) Trials of 1 episode

(b) Trials of 2 episode

Figure 7: Distribution of the total reward obtained during final episodes for trials of different lengths for experiment 2

As Figure 6 shows, letting the agent play for 2 episodes leads to generally higher reward. Although the difference is slight, in this experiment, failed episodes (failing to swing the second link high enough within 2000 timesteps) occur in the single-episode setting only.

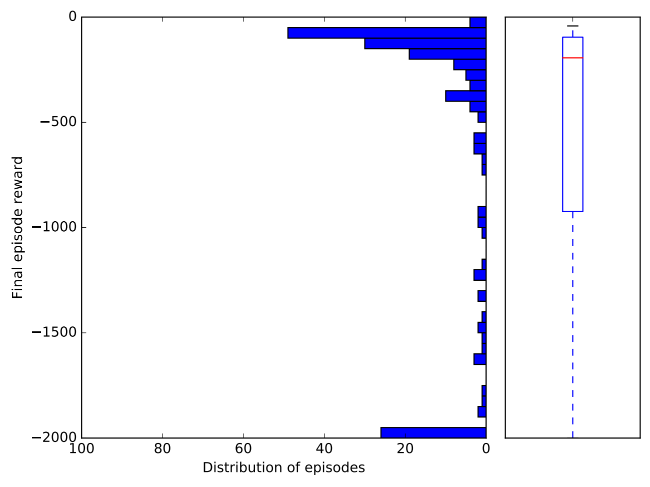
The graph of Figure 6 lacks the ”knowledge gain” information. Even if it shows that failures stop occuring, we don’t know yet exactly how much better the second episode is compared to the first one.

If we look at the distribution of reward difference between the first and second episode of dual-episode trials (Figure 4), we see that in general, although a majority of differences are negligible, second episodes improve on their respective first episodes - some of them reaching the goal as much as 1500 timesteps faster than the first episode. These are likely to be the failed first episodes followed by successful episodes after the agent gained knowledge about its environment.

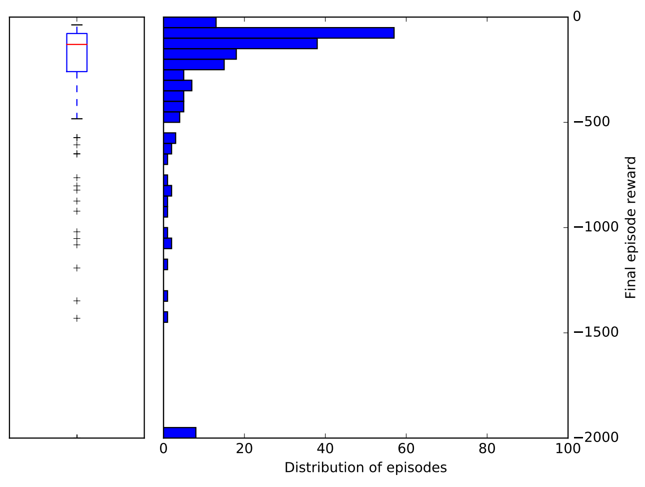
Experiment 2

One of the goals of meta reinforcement learning is to train agents that are ready to handle new and unseen situations. Testing our meta-learning agent on situations it has seen during training can give us some insight into its behaviour, but our analysis would not be complete if we did not test it in unknown situations.

In this experiment, we will try to get a grasp of the perfor-



(a) Trials of 1 episode



(b) Trials of 2 episode

Figure 8: Distribution of the total reward obtained during final episodes for trials of different lengths for experiment 3

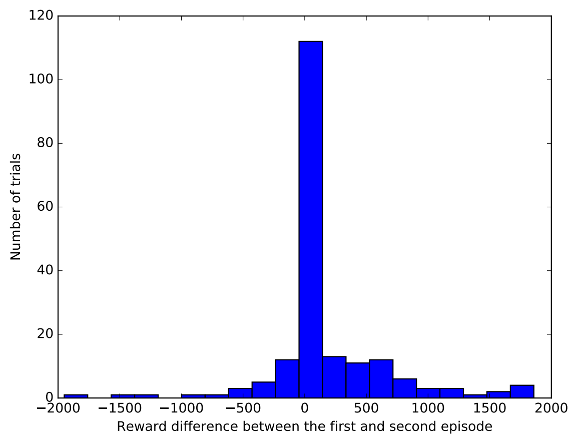


Figure 9: Distribution of the reward difference between the first and the second episode of a dual-episode trial for experiment 3

mance of our agent in completely unseen settings, where:

- the length for each pole is sampled uniformly and independently between 2 and 3: $l \sim \mathcal{U}[2, 3]$
- the mass for each pole is sampled uniformly and independently between 1 and 2: $m \sim \mathcal{U}[1, 2]$

The setting of the experiment is otherwise the same as the one presented for experiment 1.

The results however are dramatically different (see Figure 7). Instead of mostly achieving good results, both the agent having been trained in the single-episode and dual-episode settings seem to mostly fail. The agent trained in the dual-episode setting still seems to be the best performer, failing thrice less than the agent trained in the single-episode setting (37 episodes achieving a reward higher than -2000 versus 12).

Such a large number of failures could be attributed to the fact that the parameters used for this experiment are outside of the range presented to the agent during training and thus may require a wholly new strategy to be deployed in order to reach high rewards.

The difference between episode 1 and episode 2 presents a similar distribution as the one shown in experiment 1 (see Figure 5). For a majority of trials, the second episode neither improves or deteriorates reward, but there is a non-negligible group of trials in which quite a significant increase is achieved.

Experiment 3

So far, we have only tried tuning intrinsic parameters of the environment. This kept both the setting of experiment 1 and the setting of experiment 2 quite similar. In this experiment, we will explore how tampering with the observation process rather than the environment dynamics affects the performance of our agents. Hence the lengths and masses for each link are sampled uniformly and independently from the values on which the agent has been trained.

For each trial, we select at random one of the values of the observation vector which will be masked out throughout the whole trial; meaning that the observation for this value will always be 0.

Figure 8 shows the result of this experiment. Once again, even though the general performance for both the single-episode and the dual-episode setting is lower than the one presented in experiment 1, the agent trained on trials of two episodes gets the upper hand; both by having more episodes with a higer reward but also by failing 20 trials less than the agent trained on a single episode.

The distribution of reward difference between the first and second episode of dual-episode trials looks similar to the ones we have seen before: a majority of negligible differences, and visibly more gains than losses.

Conclusions

We have presented a simple reward shaping method to allow the meta-learning setup presented by Wang et al. (2016) and Duan et al. (2016) to be applied to problems with a continuous state space and a constant reward given at each timestep such as Acrobot (on which were run all the experiments of this paper) and CartPole.

We analysed the performance of a meta-learning agent and the gains it realised by playing two episodes by learning from its experience in the first episode. A distribution of Acrobot problems was generated by changing the link lenghts and masses, inherently changing the difficulty of the game and the strategy needed to be deployed to reach high rewards.

The analysis was carried on three settings:

- the training setting
- a test setting, using weight and length values which were outside of the range of values used in training
- the training setting but with a mask randomly zeroing one of the four observation values for the length of a trial.

In the three settings, playing two episodes proved to be beneficial for the agent and the reward obtained in the second episodes of trials was mostly equal or better than the reward obtained in the first episodes.

We firmly believe that the generalisation power of the meta-learning agent and its resilience to changes in the rules of the game could be further exploited by using it on more complex problems in which learning cannot occur within one episode (which was the case for some of our experiments, making the increase of performance between first and second episodes sometimes slight).

Implementation The implementation of the meta-learning A2C agent and its training process is available online and uses Tensorflow (Abadi et al., 2015) and OpenAI Gym (Brockman et al., 2016):
`https://github.com/fhennecker/meta-rl-acrobot`

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1990). Artificial neural networks. chapter Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems, pages 81–93. IEEE Press, Piscataway, NJ, USA.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2016). Rl^2: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779.

Geramifard, A., Dann, C., Klein, R. H., Dabney, W., and How, J. P. (2015). Rlpy: A value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research*, 16:1573–1578.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. (2016). ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece. IEEE. The best paper award.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Lample, G. and Chaplot, D. S. (2016). Playing FPS games with deep reinforcement learning. *CoRR*, abs/1609.05521.

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A. J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., and Hadsell, R. (2016). Learning to navigate in complex environments. *CoRR*, abs/1611.03673.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Mnih, V., Puigdomenech Badia, A., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. *ArXiv preprint arXiv:1602.01783*.

Rummery, G. A. and Niranjan, M. (1994). On-line q-learning using connectionist systems. Technical report.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016). Learning to reinforcement learn. *CoRR*, abs/1611.05763.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK.